Exploring bidirectional bounds for minimax-training of Energy-based models

Cong Geng[1], Jia Wang[2*], Li Chen[2], Zhiyong Gao[2], Jes Frellsen[3†] and Søren Hauberg[3†]

[1]China Mobile Research Institute, Beijing, 100032, China.
[2]Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China.
[3]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, Kgs. Lyngby, 2800, Denmark.

*Corresponding author(s). E-mail(s): jiawang@sjtu.edu.cn;
Contributing authors: gengcong@chinamobile.com; hilichen@sjtu.edu.cn;
zhiyong.gao@sjtu.edu.cn; jefr@dtu.dk; sohau@dtu.dk;
†These authors contributed equally to this work.

Abstract

Energy-based models (EBMs) estimate unnormalized densities in an elegant framework, but they are generally difficult to train. Recent work has linked EBMs to generative adversarial networks, by noting that they can be trained through a minimax game using a variational lower bound. To avoid the instabilities caused by minimizing a lower bound, we propose to instead work with bidirectional bounds, meaning that we maximize a lower bound and minimize an upper bound when training the EBM. We investigate four different bounds on the log-likelihood derived from different perspectives. We derive lower bounds based on the singular values of the generator Jacobian and on mutual information. To upper bound the negative log-likelihood, we consider a gradient penalty-like bound, as well as one based on diffusion processes. In all cases, we provide algorithms for evaluating the bounds. We compare the different bounds to investigate, the pros and cons of the different approaches. Finally, we demonstrate that the use of bidirectional bounds stabilizes EBM training and yields high-quality density estimation and sample generation.

Keywords: Energy-based model, density estimation, bidirectional bounds, minimax-training

1 Introduction

Generative models are growing in popularity, especially Generative Adversarial Networks (GANs) [19] and Diffusion-based models [33, 70]. However, while these can produce high-quality samples, they still have some difficulties with simultaneously explicitly estimating the data density. GANs are able to generate data samples efficiently with the generator, but the discriminator cannot be used to estimate the probability density of the data distribution, which limits their distribution-modeling ability. Diffusion-based models learn the gradients of the log-probability density function (score function) of data distribution. Then using a Langevin dynamic or ODE solver to generate samples from score functions, which may slow down or complicate sampling. Similar problems appear in density estimation using score functions.

Energy-Based Models (EBMs) are a class of probabilistic models that parameterize a distribution with an energy function, which provides the possibility for efficient density estimation. Adversarial-based EBMs try to solve the intractable normalization constant in EBMs by introducing a variational distribution so that the generator can be introduced naturally for sampling. It is this class of models we here explore.

EBMs are unnormalized log-densities with a long history in machine learning [30, 34, 65]. An EBM defines a probability distribution from the view of a Gibbs density using an energy function $E_\theta : \mathcal{X} \to \mathbb{R}$, which is parameterized by the vector θ:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}, \quad Z_\theta = \int \exp(-E_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}, \tag{1}$$

where Z_θ is the normalization constant (also known as the partition function). This construction can, in principle, model any density with a suitable choice of E_θ. We typically learn EBMs using maximum likelihood estimation (MLE), where we seek a value of θ that minimizes the *negative* data log-likelihood:

$$\begin{aligned} L(\theta) &:= -\,\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[E_\theta(\mathbf{x})] + \log Z_\theta, \end{aligned} \tag{2}$$

where p_{data} is the data generating distribution.

The key challenge with EBMs is the lack of a closed-form expression for the normalization constant, Z_θ, for non-standard energy functions. This hinders pointwise density evaluation, learning, and sampling. These tasks are, therefore, often performed using approximate algorithms such as Markov Chain Monte Carlo (MCMC) [50, 52].

1.1 Minimax games on bounds

Recently, a series of papers [2, 8, 40, 44] have linked EBMs and *Wasserstein GANs* (WGANs) [5], such a minimax game forms an approximate learning strategy for EBMs. We here repeat this development following Grathwohl et al. [20]. First, we develop a lower bound for the log-normalization constant using a proposal (variational) distribution $p_g(\mathbf{x})$ followed by Jensen's inequality:

$$\log Z_\theta = \log \mathbb{E}_{\mathbf{x}\sim p_g}\left[\frac{\exp(-E_\theta(\mathbf{x}))}{p_g(\mathbf{x})}\right] \tag{3}$$

$$\geq \mathbb{E}_{\mathbf{x}\sim p_g}\left[\log \frac{\exp(-E_\theta(\mathbf{x}))}{p_g(\mathbf{x})}\right] \tag{4}$$

$$= -\mathbb{E}_{\mathbf{x}\sim p_g}[E_\theta(\mathbf{x})] + H[p_g]. \tag{5}$$

Substituting this into Eq. (2), we can lower bound the negative log-likelihood as follows:

$$\begin{aligned} &L(\theta) \geq \mathcal{L}(\theta) := \\ &\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[E_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[E_\theta(\mathbf{x})] + H[p_g]. \end{aligned} \tag{6}$$

We emphasize that this bound is tight when $p_g = p_\theta$, which is attained when $\mathcal{L}(\theta)$ is maximized with respect to p_g. When the bound is tight, we can perform maximum likelihood estimation of θ by minimizing $\mathcal{L}(\theta)$. Accordingly, training of an EBM can be seen as a minimax game:

$$\begin{aligned} &\min_{E_\theta}\max_{p_g}\{\mathcal{L}(\theta)\} = \\ &\min_{E_\theta}\max_{p_g}\big\{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[E_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[E_\theta(\mathbf{x})] \\ &\qquad\qquad + H[p_g]\big\}. \end{aligned} \tag{7}$$

Assuming we can evaluate the entropy of the proposal, we thus have a tractable and elegant way of learning EBMs.

The above analysis, however, assumes that we can tighten the bound by finding a proposal distribution p_g that exactly samples from p_θ, i.e. that p_θ is within the variational family. In practice, it is, however, difficult to tighten the lower bound during the training of p_g. As illustrated in Figure 2, if we replace the energy function with the true negative log-likelihood of p_{data}, and optimize p_g to tighten the bound, we observe clear mode collapse. We also calculate $\mathrm{KL}\,(p_g \| p_{data})$, which is always notably different from zero. This explains why the bound is loose. When the bound is *not* tight, as one would generally expect during training, we note that $\mathcal{L}(\theta)$ is a *lower bound* on the negative log-likelihood. This is the opposite of conventional *variational inference* [6], where a lower bound on the (non-negative) log-likelihood is maximized. In general, minimizing a lower bound introduces a risk of training towards parameters, where the bound is

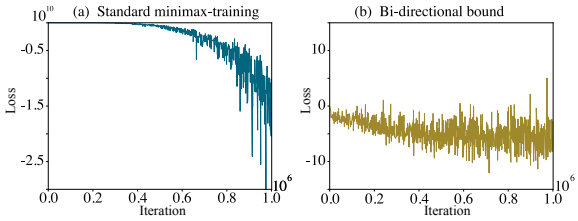


Fig. 1 Loss function with standard minimax-training (a) or with one of the alternative bounds (b).

loosest, instead of where the true objective is near-optimal. In particular, a minimum of the lower bound may be $-\infty$, which is rather unhelpful. To see if this is a problem *in practice*, we fit an EBM to toy data by minimax training using Eq. (7). Figure 1 shows that the energy of the training data gradually diverges towards the unhelpful $-\infty$ as training proceeds, suggesting that the potential issues with minimizing a lower bound do appear in practice. In contrast, Fig. 1b shows the training curve associated with one of our proposed bounds (23), where it is evident that the divergent training is avoided.

In this paper, we investigate the issues caused by minimizing a lower bound. We put forward the hypothesis that it is better to minimize an upper bound and maximize a lower bound (Sec. 2), rather than playing a minimax game on a single bound. We propose a collection of such *bidirectional bounds* (Secs. 2.1 to 2.4) and compare them empirically (Sec. 4). We demonstrate that this approach surpasses or matches state-of-the-art on diverse tasks at a negligible computational cost. We position our work to previous ones and touch on historical developments in Sec. 3. We note that this paper extends a previous conference paper [17], where we first proposed the idea of using bidirectional bounds. Here we consider more bounds and perform the associated empirical study.

2 Minimax-training with bidirectional bounds

Our main hypothesis is that instead of solving a minimax game with a lower bounded value function (6), it is more meaningful to bound $L(\theta)$ from both above and below:

$$\lfloor L(\theta) \rfloor \leq L(\theta) \leq \lceil L(\theta) \rceil. \tag{8}$$

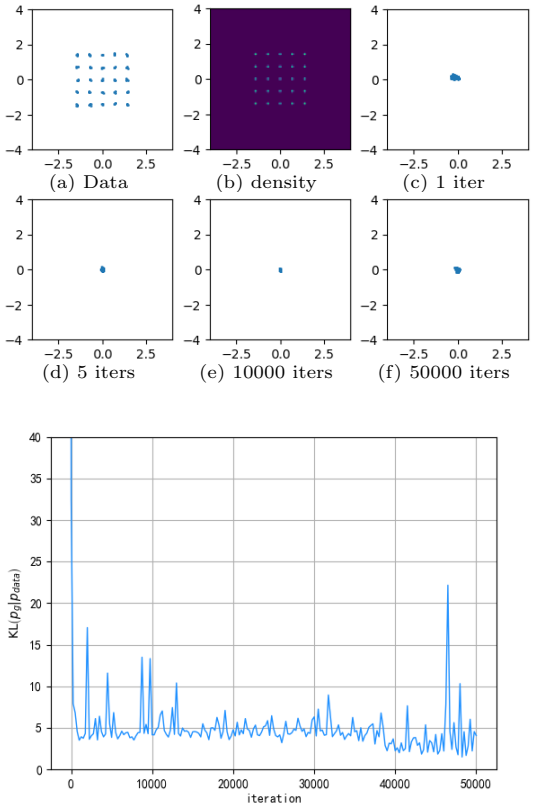


Fig. 2 Visualization of generation and plot curve of $\mathrm{KL}\,(p_g \| p_{data})$ with different iters when fixing the energy function to be the real log-likelihood.

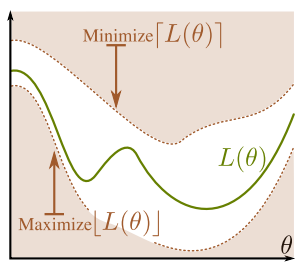


Fig. 3 The bidirectional bounds 'sandwich' the negative data log-likelihood, which results in stable training.

From this bidirectional bound, we can follow an alternating optimization approach as follows:

1. minimize $\lceil L(\theta) \rceil$ with respect to E_θ.
2. maximize $\lfloor L(\theta) \rfloor$ with respect to p_g.

This 'sandwiching' is illustrated in Fig. 3. We hypothesize that this approach avoids the issues of minimizing a lower bound. In this paper, we let $\lfloor L(\theta) \rfloor = \mathcal{L}(\theta)$, and propose two approximations of this term. Further, we let $\lceil L(\theta) \rceil =$

$\mathcal{L}(\theta) + \lceil \mathrm{KL}\left(p_g \| p_\theta\right) \rceil$, where $\lceil \mathrm{KL}\left(p_g \| p_\theta\right) \rceil$ denotes an upper bound of $\mathrm{KL}\left(p_g \| p_\theta\right)$. We use two different methods to find this bound. Below the four different bounds are discussed in detail.

2.1 A lower bound from singular values

Eq. (6) provides a lower bound $\mathcal{L}(\theta)$ on $L(\theta)$. The first two terms of the bound can be evaluated easily through samples from p_{data} and p_g, respectively. Evaluating the entropy of p_g is not as straightforward. Following common practice in generative adversarial networks (GANs) [19], we define p_g from a base variable $\mathbf{z} \sim p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is transformed by a 'generator' network $G : \mathbb{R}^d \to \mathbb{R}^D$, to which we add further noise [1]. Specifically, sampling from p_g becomes

$$\begin{aligned} &\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &\mathbf{x} = G(\mathbf{z}) + \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2_{\text{noise}}\mathbf{I}), \end{aligned} \tag{9}$$

where σ^2_{noise} denotes the variance scale of ϵ. The generator entropy is, thus, bounded by the entropy of the noise-free sampler, i.e. $H[p_g] \geq H[G(\mathbf{z})]$. The variable $G(\mathbf{z})$ has density over a d-dimensional manifold[2] in $\mathbb{R}^D$, which, following standard change-of-variables, is

$$\log p(G(\mathbf{z})) = \log p_0(\mathbf{z}) - \frac{1}{2}\log\det(\mathbf{J}_\mathbf{z}^\intercal \mathbf{J}_\mathbf{z}), \tag{10}$$

where $\mathbf{J}_\mathbf{z} \in \mathbb{R}^{D\times d}$ is the Jacobian of G at $\mathbf{z}$. This assumes that the Jacobian exists, i.e. that the generator G is differentiable. The entropy of p_g is then bounded from below by

$$\begin{aligned} H[p_g] &= -\mathbb{E}_{\mathbf{x}\sim p_g}\left[\log p_g(\mathbf{x})\right] &(11)\\ &\geq H[p_0] + \mathbb{E}_{\mathbf{z}\sim p_0}\left[\frac{1}{2}\log\det(\mathbf{J}_\mathbf{z}^\intercal \mathbf{J}_\mathbf{z})\right]. &(12)\end{aligned}$$

[1] Refer to Sec. A for the rationale behind the design of p_g.

[2] Note that Eq. (10) is only the density *on* the spanned manifold, and that the density is zero *off* the manifold.

The entropy of p_0 has closed-form expression $H[p_0] = {}^d\!/_2(1+\log(2\pi))$. To avoid the computationally expensive evaluation of the log-determinant-term, we bound this further from below:

$$\frac{1}{2}\log\det(\mathbf{J}_\mathbf{z}^\intercal \mathbf{J}_\mathbf{z}) = \frac{1}{2}\sum_{i=1}^{d}\log s_i^2 \geq d\log s_1, \tag{13}$$

where $s_d \geq \ldots \geq s_1$ are the singular values of the Jacobian $\mathbf{J}_\mathbf{z}$. We emphasize that this bound is only tight when all singular values are identical. Combining these developments gives our final lower bound:

$$\begin{aligned} L(\theta) \geq \lfloor L(\theta) \rfloor = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[E_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[E_\theta(\mathbf{x})] \\ + H[p_0] + \mathbb{E}_{\mathbf{z}\sim p_0}\left[d\log s_1(\mathbf{z})\right]. \end{aligned} \tag{14}$$

Evaluating the lower bound. The lower bound in Eq. (14) requires access to the smallest singular value of the Jacobian $\mathbf{J} = \partial_\mathbf{z} G(\mathbf{z})$. Note that this singular value satisfy $s_1 = \|\mathbf{J}\mathbf{v}_{\min}\|_2 = \min_{\mathbf{v}\neq\mathbf{0}} \frac{\|\mathbf{J}\mathbf{v}\|_2}{\|\mathbf{v}\|_2}$. This singular value $\mathbf{v}_{\min}$ can then be found through optimization, where we choose to use the well-known single-vector *LOPCG algorithm* [41]. This algorithm iteratively minimizes the generalized Rayleigh quotient,

$$\rho(\mathbf{v}) := \frac{\mathbf{v}^\intercal \mathbf{J}^\intercal \mathbf{J}\mathbf{v}}{\mathbf{v}^\intercal \mathbf{v}}, \tag{15}$$

which converges to $\mathbf{v}_{\min}$. The gradient of $\rho(\mathbf{v})$ is proportional to $r = \mathbf{J}^\intercal\mathbf{J}\mathbf{v} - \rho(\mathbf{v})\mathbf{v}$. We use Jacobian-vector products to avoid explicitly forming the Jacobian $\mathbf{J}$ as these can be efficiently using automatic differentiation. To compute $\mathbf{J}^\intercal\mathbf{J}\mathbf{v}$, we use a neat trick, which we here specify in `pytorch`-notation:

$$\mathbf{J}^\intercal\mathbf{J}\mathbf{v} = ((\mathbf{J}\mathbf{v})^\intercal\mathbf{J})^\intercal = \nabla_\mathbf{z}\left((\mathbf{J}\mathbf{v})^\intercal\texttt{.detach()}\cdot G(\mathbf{z})\right)^\intercal. \tag{16}$$

The optimal step size for this iterative optimization can be found by maximizing the Rayleigh quotient (15). Furthermore, we follow the suggestions of Knyazev [41] which improve numerical stability and accelerate convergence. For brevity, we omit these.

Computational cost. The smallest singular value of Jacobian is obtained by an iterative inner loop, which can be costly. For each iteration,

we need to compute $\mathbf{Jv}$ *four* times and $\mathbf{J}^\intercal\mathbf{Jv}$ *one* time. Two matrix decompositions are also needed for a $d \times 3$ matrix and $D \times 3$ matrix respectively. Note that we need not differentiate through these operations, which hence are time-consuming but memory efficient. We cannot *priorly* estimate the needed number of iterations of the conjugate gradient (CG) method[3], but empirically we have found the estimator to be insensitive to the number iterations. Most likely, this is because we only use the singular value to bound the entropy. In practice, we start from a Hutchinson estimator [35] which is an upper bound of the entropy, and then use very few iterations to reduce the estimate to a lower bound. We will empirically investigate this in Fig. 6.

2.2 A lower bound from mutual information

An alternative estimator of the entropy $H[p_g]$ can be derived through links to mutual information [44]. Following Kumar et al. [44], the generator (9) entropy can be expressed in terms of the mutual information between $\mathbf{x}$ and $\mathbf{z}$ as

$$I[\mathbf{x}, \mathbf{z}] = H[\mathbf{x}] - H[\mathbf{x} \mid \mathbf{z}]. \tag{17}$$

We note that $H[\mathbf{x} \mid \mathbf{z}] = \frac{1}{2}\log(2\pi e\sigma^2_{\text{noise}})$ is constant. The mutual information $I(\mathbf{x}, \mathbf{z})$ equals the KL divergence between the product of marginals $p_g(\mathbf{x})p(\mathbf{z})$ and the joint distribution $p(\mathbf{x}, \mathbf{z})$. Rather than estimating this KL divergence, we follow Kumar et al. [44] in using the estimator from Hjelm et al. [32], which computes the Jensen-Shannon divergence between $p_g(\mathbf{x})p(\mathbf{z})$ and $p(\mathbf{x}, \mathbf{z})$. This choice is based on empirical observations by Kumar et al. [44] that this divergence works better in practice, albeit without implying a tight bound. The estimator then becomes

$$\begin{aligned} I_{JSD}(\mathbf{x}, \mathbf{z}) = \sup_{T \in \mathcal{T}} \mathbb{E}_{p(\mathbf{x},\mathbf{z})}[-\operatorname{sp}(-T(\mathbf{x}, \mathbf{z}))] \\ - \mathbb{E}_{p_g(\mathbf{x})p(\mathbf{z})}[\operatorname{sp}(T(\mathbf{x}, \mathbf{z}))], \end{aligned} \tag{18}$$

where $\operatorname{sp}(\cdot)$ is the softplus function, and T is an additional discriminator represented by a neural network. As a minor side-remark, we note that Kumar et al. [44] consider a deterministic generator, i.e. $\mathbf{x} = G(\mathbf{z})$, while we add further Gaussian noise. In the case of a deterministic generator, $H(\mathbf{x}|\mathbf{z}) = -\infty \neq 0$, while the stochastic generator does not have such degeneracy.

[3] https://web.cs.ucdavis.edu/ bai/ECS231/Slides/cg_eig.pdf

2.3 An upper bound from gradient penalties

As a first upper bound for the log-likelihood, we take inspiration from how WGANs [5] are trained as there are strong ties between our lower bound (6) and the WGAN objective,

$$L_{\text{WGAN}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[E_\theta(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim p_g}\left[E_\theta(\mathbf{x})\right]. \tag{19}$$

We note that the only difference between Eq. (19) and Eq. (6) is the entropy term. We can, thus, expect lessons from WGANs to carry over to the EBM setting. The original WGAN paper [5] noted that *gradient clipping* was needed to avoid optimization instabilities. Gulrajani et al. [22] proposed to instead add a *gradient penalty* to the loss, such that it becomes

$$\begin{aligned} L_{\text{WGAN-GP}} = L_{\text{WGAN}} \\ + \lambda\mathbb{E}_{\hat{\mathbf{x}}\sim g(\mathbf{x})}\left[(\|\nabla_{\hat{\mathbf{x}}}E_\theta(\hat{\mathbf{x}})\|_2 - 1)^2\right], \end{aligned} \tag{20}$$

where $\hat{\mathbf{x}}$ is a uniformly sampled interpolation between real data and generated samples. This is known to be a good way to train WGANs [22]. We note that this loss is similar to our lower bound, but that the gradient penalty adds a further positive term. One could, thus, speculate that the result is an upper bound. From this intuition, we have proved the following theorem in the conference version of the paper [17].

Theorem 1 *Suppose $f : \mathcal{X} \to \mathbb{R}$ is L-Lipschitz continuous, $g(x)$ is a probability density function with finite support, then there exist constants $M, m \geq 0$ and $p \geq 1$ such that:*

$$\begin{aligned} \log \mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\log f(\mathbf{x})\right] \\ \leq M(\mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log f(\mathbf{x})\|^p\right] + m)^{1/p}. \end{aligned} \tag{21}$$

Note that the bound in Eq. (21) can be simplified by dropping the power $1/p$ if $\mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log f(\mathbf{x})\|^p\right] + m \geq 1$,

$$\begin{aligned} \log\left[\mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[f(\mathbf{x})\right]\right] - \mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\log f(\mathbf{x})\right] \\ \leq M\mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log f(\mathbf{x})\|^p\right] + Mm. \end{aligned} \tag{22}$$

In our EBM, we set $f(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{p_g(\mathbf{x})}$ and $g(\mathbf{x}) = p_g(\mathbf{x})$. We note that $\log\left[\mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[f(\mathbf{x})\right]\right] - \mathbb{E}_{\mathbf{x}\sim g(\mathbf{x})}\left[\log f(\mathbf{x})\right] = \mathrm{KL}\left(p_g \| p_\theta\right)$, which means that Eq. (22) is an upper bound of $\mathrm{KL}\left(p_g\|p_\theta\right)$. We then get an upper bound as $\lceil L(\theta)\rceil = \mathcal{L}(\theta) + \lceil \mathrm{KL}\left(p_g\|p_\theta\right)\rceil$.

Empirically, we have found that the bound simplification in Eq. (22) holds throughout most of the training, and therefore use the following (approximate) upper bound:

$$\begin{aligned}\lceil L(\theta)\rceil &= \mathcal{L}(\theta) + Mm \\ &+ M\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) + \nabla_{\mathbf{x}}\log p_g(\mathbf{x})\|^p\right].\end{aligned} \tag{23}$$

For a specific choice of $f(\mathbf{x})$, we can view m as a constant, and the term Mm can be ignored during optimization. This developed upper bound has a nice interpretation as the lower bound in (6) plus a gradient penalty. This penalty, however, has a different form than the classic WGAN penalty, which is developed from a regularization perspective. Our proposed upper bound can, thus, provide justification for the commonly used regularization from a maximum likelihood perspective.

Bound tightness. When $p_\theta = p_g$, we see that $\|\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) + \nabla_{\mathbf{x}}\log p_g(\mathbf{x})\|^p = 0$. In Theorem 1, m is a constant related to the Lipschitz constant of $\log f(\mathbf{x})$ which satisfy $\|\nabla_{\mathbf{x}}\log f(\mathbf{x})\|^p \leq \|\nabla_{\mathbf{y}}\log f(\mathbf{y})\|^p + m$ for all $\mathbf{x}, \mathbf{y}$ (see details in the proof). When $p_\theta = p_g$ we furthermore have that $\|\nabla_{\mathbf{x}}\log f(\mathbf{x})\|^p = 0$, such that $m = 0$. The upper bound then reduces to $\lceil L(\theta)\rceil = \mathcal{L}(\theta) = L(\theta)$, which implies that the bound is tight.

Evaluating the upper bound. Evaluating Eq. (23) requires solving two challenges. First, we have to compute $\nabla_{\mathbf{x}}\log p_g(\mathbf{x})$. Empirically we have found that current methods [47, 64] are not sufficiently efficient to be practical. Instead, we propose to approximate the term $\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) + \nabla_{\mathbf{x}}\log p_g(\mathbf{x})\|^p\right]$ by further loosening the bound

$$\begin{aligned}&\|\nabla_{G(\mathbf{z})} E_\theta(G(\mathbf{z})) + \nabla_{G(\mathbf{z})}\log p_g(G(\mathbf{z}))\|^p \\ &\leq \frac{\|\nabla_{G(\mathbf{z})} E_\theta(G(\mathbf{z}))\mathbf{J}_{\mathbf{z}} + \nabla_{G(\mathbf{z})}\log p_g(G(\mathbf{z}))\mathbf{J}_{\mathbf{z}}\|^p}{s_1^p} \\ &\leq \frac{\|\nabla_{G(\mathbf{z})} E_\theta(G(\mathbf{z}))\mathbf{J}_{\mathbf{z}} + \nabla_{\mathbf{z}}\log p_g(G(\mathbf{z}))\|^p}{s_1^p}.\end{aligned} \tag{24}$$

Here s_1 is the smallest singular value of $\mathbf{J}_{\mathbf{z}}$. In the case $p = 2$, we can use Hutchinson's estimator [35]:

$$\begin{aligned}&\|\nabla_{\mathbf{x}} E_\theta(\mathbf{x})\mathbf{J}_{\mathbf{z}} + \nabla_{\mathbf{z}}\log p_g(G(\mathbf{z}))\|_2^2 \\ &= \mathbb{E}_{\mathbf{v}}\left[\left(\nabla_{\mathbf{x}} E_\theta(\mathbf{x})\mathbf{J}_{\mathbf{z}}\mathbf{v} + \nabla_{\mathbf{z}}\log p_g(G(\mathbf{z}))\mathbf{v}\right)^2\right],\end{aligned} \tag{25}$$

where $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. This can be evaluated efficiently using automatic differentiation.

The second challenge is to compute $\log p_g(\mathbf{x})$ which amounts to computing the log-determinant-term $\log\det(\mathbf{J}_{\mathbf{z}}^\intercal\mathbf{J}_{\mathbf{z}})$ of the generator $G(\mathbf{z})$ as dictated by Eq. (10). Here, we choose to use our entropy estimator (13).

2.4 An upper bound from diffusion

As an alternative upper bound of $\mathrm{KL}\left(p_g\|p_\theta\right)$, we take inspiration from Song et al. [69] and introduce a time t to build a diffusion model. If we employ a stochastic differential equation (SDE) to diffuse a distribution $p(\mathbf{x})$ towards a noise distribution, we get an SDE of the form:

$$\mathrm{d}\mathbf{x} = \boldsymbol{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}. \tag{26}$$

Here we choose the Variance Exploding (VE) SDE [70] for simplicity, which means $\boldsymbol{f}(\mathbf{x}, t) = 0$, and $g(t) = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^t\sqrt{2\log\frac{\sigma_{\max}}{\sigma_{\min}}}$. The solution of this SDE is a diffusion process$\{\mathbf{x}(t)\}_{t\in[0,T]}$, where $[0, T]$ is a fixed time horizon. We let $p_t(\mathbf{x})$ denote the marginal distribution of $\{\mathbf{x}(t)\}$, and $p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right)$ denote the transition distribution from $\mathbf{x}(0)$ to $\mathbf{x}(t)$. Finally, $p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right)$ is a Gaussian distribution where both mean and variance can be computed. If we diffuse $p_g(\mathbf{x})$ and $p_\theta(\mathbf{x})$ with the same noise distribution, then we can compute an upper bound of the KL divergence [69]

$$\begin{aligned}&\mathrm{KL}(p_g(\mathbf{x})\|p_\theta(\mathbf{x})) \\ &= \mathrm{KL}\left(p_{gT}(\mathbf{x})\|p_{\theta T}(\mathbf{x})\right) - \int_0^T \frac{\partial\mathrm{KL}\left(p_{gt}(\mathbf{x})\|p_{\theta t}(\mathbf{x})\right)}{\partial t}\mathrm{d}t \\ &\leq \frac{1}{2}\int_0^T \mathbb{E}_{p_{gt}(\mathbf{x})}\left[g^2(t)\left\|\nabla_{\mathbf{x}}\log p_{gt}\left(\mathbf{x}\right) - \nabla_{\mathbf{x}}\log p_{\theta t}(\mathbf{x})\right\|_2^2\right]\mathrm{d}t \\ &+ \mathrm{KL}\left(p_{gT}(\mathbf{x})\|p_{\theta T}(\mathbf{x})\right).\end{aligned} \tag{27}$$

If we introduce time t into the energy function, i.e. $p_{\theta t}(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x},t))}{Z_\theta(t)}$, then $p_\theta(\mathbf{x}) =$

$\frac{\exp(-E_\theta(\mathbf{x},0))}{Z_\theta(0)}$. Combined with score matching [74], we immediately obtain:

$$\begin{aligned}\mathrm{KL}\left(p_g(\mathbf{x})\|p_\theta(\mathbf{x})\right) &\leq \mathrm{KL}\left(p_{gT}(\mathbf{x})\|p_{\theta T}(\mathbf{x})\right) \\ &+ \frac{1}{2}\int_0^T \mathbb{E}_{p(\mathbf{x})p_{0t}(\mathbf{x}'|\mathbf{x})}\Big[g^2(t)\big\|\nabla_{\mathbf{x}'}\log p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right) \\ &\qquad - E'_\theta(\mathbf{x},t)\big\|_2^2\Big]\mathrm{d}t + C,\end{aligned} \tag{28}$$

where C is a constant independent of θ. Note that for this to be a valid bound $p_{\theta t}(\mathbf{x})$ needs to satisfy Fokker–Planck equation [72],

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \nabla_{\mathbf{x}} \cdot \left(\frac{1}{2}g^2(t)p_t(\mathbf{x})\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right). \tag{29}$$

When we directly represent $E_\theta(\mathbf{x},t)$ with a neural network parameterized by θ and t, it is difficult to ensure that the constraint is satisfied, so we have no strict guarantee that the bound holds. Following Song and Ermon [68], we let $E_\theta(\mathbf{x},t) = \frac{E_\theta(\mathbf{x})}{\sigma(t)}$, where $\sigma(t)$ denotes the variance of the transition distribution $p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right)$. Song and Ermon [68] reports that this approximation works well in practice.

Altogether this yields our diffusion-based upper bound

$$\begin{aligned}\lceil L(\theta)\rceil &= \mathcal{L}(\theta) + \lceil \mathrm{KL}\left(p_g\|p_\theta\right)\rceil = \mathcal{L}(\theta) \\ &+ \frac{1}{2}\int_0^T \mathbb{E}_{p(\mathbf{x})p_{0t}(\mathbf{x}'|\mathbf{x})}\big[g^2(t)\big\|\nabla_{\mathbf{x}'}\log p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right) \\ &\qquad - E'_\theta(\mathbf{x},t)\big\|_2^2\big]\mathrm{d}t \\ &+ \mathrm{KL}\left(p_{gT}(\mathbf{x})\|p_{\theta T}(\mathbf{x})\right) + C.\end{aligned} \tag{30}$$

The term $\mathrm{KL}\left(p_{gT}\|p_{\theta T}\right)$ is close to 0, if we add enough noise, and we practically ignore the term. The constant C can be ignored during training. The integral is computed by calculating the following expectation:

$$\begin{aligned}\frac{1}{T}\int_0^T p(t)\mathbb{E}_{p(\mathbf{x})p_{0t}(\mathbf{x}'|\mathbf{x})}\big[g^2(t)\big\|\nabla_{\mathbf{x}'}\log p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right) \\ - E'_\theta(\mathbf{x},t)\big\|_2^2\big]\mathrm{d}t,\end{aligned} \tag{31}$$

where $p(t)$ is uniform over $[0,T]$. Computationally, we sample $t \sim p(t)$ to evaluate Eq. (31) using Monte Carlo. Since $\log p_{0t}\left(\mathbf{x}' \mid \mathbf{x}\right)$ is Gaussian, the computation of Eq. (31) is efficient [69].

Bound tightness. If p_g and p_θ have finite second moments and continuous second-order derivatives, then our bound is tight [69]. Furthermore, if $p_g = p_\theta$, then Eq. (27) is easily seen to 0 such that $\lceil L(\theta)\rceil = \mathcal{L}(\theta) = L(\theta)$ and our bound becomes tight.

2.5 Relevance for Cooperative Networks

Cooperative Networks (CoopNets) employ cooperative training to train both the generator and energy function. To optimize the energy function, CoopNets initializes samples from p_θ using the generator's output, which significantly accelerates MCMC sampling. As mentioned by [83], let $\mathcal{M}_\theta^{(k)}$ denote k steps MCMC transition kernel, then $\mathcal{M}_\theta^{(k)}p_g$ is the approximated distribution of p_θ after MCMC refinement. Therefore, the energy training objective is

$$\mathrm{KL}\left(p_{\mathrm{data}}\|p_\theta\right) - \mathrm{KL}\left(\mathcal{M}_\theta^{(k)}p_g\|p_\theta\right). \tag{32}$$

We see that CoopNets still minimize a lower bound, even though the difference between this bound and $\mathrm{KL}\left(p_{\mathrm{data}}\|p_\theta\right)$ is tighter than that between $\mathcal{L}(\theta)$ and $L(\theta)$. When optimizing the generator, we minimize $\mathrm{KL}(p_g\|p_\theta)$, while CoopNets minimize $\mathrm{KL}(\mathcal{M}_\theta^{(k)}p_g\|p_g)$. The CoopNet objective may have some advantages regarding mode collapse, but it needs MCMC sampling from the posterior $p_g(z \mid x)$, which can be costly and will depend on the settings of MCMC sampling.

3 Related work

3.1 Deep generative EBMs

In machine learning, interests in EBMs can be dated back to Hopfield networks [34], Boltzmann machines [3, 30, 55] and restricted Boltzmann machines [29, 65], see e.g. reviews in [45] and [63]. It is difficult to learn and evaluate these models because of the expensive computation of the normalization constant. Some MLE-based learning algorithms use expensive MCMC-based methods to estimate the gradient or the normalization constant, see e.g. [7, 15, 21, 61]. Moreover, some

non-deep EBMs also employed MCMC-based learning on texture modeling [87, 88] and natural image patterns [78, 79]. To avoid the costly MCMC approximation, Nijkamp et al. [53] proposed a short-run MCMC procedure for EBMs and later described an interesting way to check whether EBM is well-learned with MCMC [54]. Hinton [29] proposed a k-step Contrastive Divergence (CD-k) method to approximate the negative phase log-likelihood gradient. Hyvärinen [36] proposed an alternative method that used score matching to train non-normalized graphical models. Subsequently, deep EBMs such as deep belief networks [31] and deep Boltzmann machines [60], have been proposed.

Recently, deep generative EBMs have grown in popularity, especially for image generation. Grathwohl et al. [20] outlined three classes of learning-based methods for their current developments and obvious weaknesses, which we summarize here: (1) MLE methods with MCMC sampling are slow while (2) score-matching-based methods are comparable faster but do not work with discontinuous nonlinearities, both of them are unstable. (3) Noise-contrastive estimation [23] relies on density ratio estimation to learn energy functions but typically does not scale well with the data dimensionality.

Our proposed method belongs to a fourth class of methods that uses a simultaneously learned generator or variational distribution to avoid costly MCMC sampling. For this class, Kim and Bengio [40] also proposed to use a generator function and update the generator using the same lower bound as us, but a different entropy approximation. Furthermore, without an extra regularizer like our upper bound when optimizing the energy function, their energy function needs to be explicitly designed to prevent it from growing to infinity which limits its potential.

3.2 Adversarial-based EBMs

In the past years, generative models based on adversarial training have made significant gains through variations of the GAN [19]. Unlike EBMs, these GAN-style models are not explicit likelihood models. It is worth noting that the only difference between the WGAN loss [5] and the lower bound in Eq. (6) is the entropy term. We can, thus, expect lessons from the WGAN to apply to EBMs as well. Xie et al. [80, 81] and Wu et al. [76] were among the first to employ adversarial training of EBMs. Zhai et al. [86] proposed to use the same lower bound as us with a minimax game to jointly optimize the energy function and generator. However, a specifically designed bounded multi-modal energy function is needed for their methods which limits its potential. Furthermore, their methods provide no theoretical guarantees on the approximation quality of the generator entropy, and their formulation does not include an upper bound on the regularization of the energy function.

Dai et al. [11] also proposed to jointly learn the energy function and a generator using an adversarial learning framework. Two approaches were employed to maximize the generator entropy: One is to maximize the entropy by minimizing the conditional entropy using a variational upper bound, and another is to assume the data is isotropic Gaussian, which is not suitable for high-dimensional data. Kumar et al. [44] also consider adversarial learning with an estimator of the entropy of the generator that is based on mutual information. Our lower bound from mutual information (18) is nearly identical to the estimator from Kumar et al. [44], with the only difference being that our generator is stochastic. Abbasnejad et al. [1] approximated the entropy using the generator function's Jacobian log-determinant. However, their method does not scale to high-dimensional data. Han et al. [24] proposed a divergence triangle loss with an adversarial learning strategy, whose training mechanism is quite different from ours. An extra encoder is also needed for learning the generator.

A *cooperative learning* between the energy function and a generator was proposed by Xie et al. [82, 83, 84, 85]. However, this type of approach relies on Langevin dynamics or MCMC to draw samples from the EBM, which can be expensive and difficult to tune. Grathwohl et al. [20] proposed VERA to avoid the use of MCMC which is similar to our work. But unlike us, VERA plays a minimax game both on a lower bound and approximates the gradient of the entropy term using variational inference. Furthermore, VERA is memory-consuming and uses a zero-centered gradient penalty as a regularizer for the energy function, which is a heuristic without an explicit theoretical guarantee.

4 Experiments

To demonstrate the efficiency of training EBMs with bidirectional bounds, we compare against a selection of methods for different model classes. To compare against GANs, we use *deep convolutional GANs (DCGANs)* by Radford et al. [57], *spectrally normalized GANS (SNGANs)* by Miyato et al. [51] and the *WGAN with zero-centered gradient penalty (WGAN-0GP)* by Thanh-Tung et al. [73]. To compare against other EBMs, we use the *maximum entropy generators (MEG)* by Kumar et al. [44] and the *variational entropy regularized approximate maximum likelihood (VERA)* estimator [20]. To compare against CoopNets, we use two similar methods [82, 84] and *EBM-VAE* [85]. We also consider *DDPM* [33] and *NCSN* [67] to represent diffusion-based models and score-matching models, respectively.

For bidirectional bounds, we consider all four combinations of our proposed bounds to determine which works best in practice. To differentiate the combinations, we denote

- SV as the lower bound derived from singular values (14),
- MI as the mutual information lower bound (18),
- GP as the gradient penalties upper bound (23), and
- diff as the score-based diffusion upper bound (28).

We then denote combinations using $\text{EBM}_{\text{lower+upper}}$, e.g. $\text{EBM}_{\text{SV+GP}}$. It is worth noting that both the lower bound based on singular values and the upper bound based on gradient penalties require computing $H(p_g)$. Since this is computationally expensive, we generally recommend using these bounds jointly in order to reduce the total computational cost. To get further insight into the influence of using an upper bound, we introduce another baseline in which we use a zero-centered gradient penalty instead of an actual upper bound. This is similar in spirit to our GP upper bound. We denote this model EBM_{0GP}.

4.1 Training details

On the small toy data, we realize all models using multi-layer perceptrons (MLPs), while for challenging datasets, we use the same convolutional architecture as `StudioGAN` [38]. The discriminator network T in mutual information-based lower bound (18) follows the design of Kumar et al. [44]. We conduct all experiments on a single 12GB NVIDIA Titan GPU using `pytorch` [56]. We found that training is improved when we use a positive margin for our energy function of gradient penalties upper bound to balance the terms. Specifically,

$$\begin{aligned}\lceil L(\theta) \rceil &= \lfloor L(\theta) \rfloor \\ &\quad + \left[M \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})} \left[\|\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_g(\mathbf{x})\|^p \right] - \zeta \right]_+,\end{aligned} \tag{33}$$

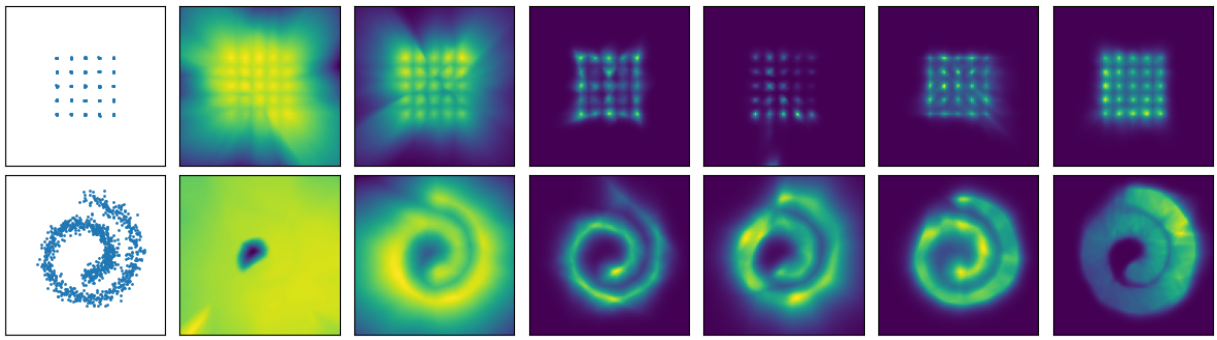
where $[\cdot]_+ = \max(0, \cdot)$ is the usual hinge. In most experiments, we set $\zeta = 1$ as this lets us use the simplified bound in Eq. (23). Generally, we recommend starting with $\zeta = 0$ and successively considering larger values until training stabilizes. For $\text{EBM}_{\text{SV+GP}}$, like [12, 20], we also find it helpful to add an entropy regularizer weight λ, which is consistent with theory when set to 1. In larger-scale experiments, we reduced the weight of the entropy regularizer to improve performance. For $\text{EBM}_{\text{MI+diff}}$, we choose the time horizon $[0, T]$ for diffusion process as [0,1]. $\sigma_{min} = 0.01$ and σ_{max} is adjusted to different experiments. But we set $\sigma_{max} = 0.1$ for most experiments.

4.2 Toy data

We first experiment with the 25-Gaussians and swiss-roll datasets, and Fig. 4 shows estimated densities using our methods, WGAN-0GP, and two baselines. We observe that WGAN's discriminator does not provide a useful density estimate. This is unsurprising as WGAN is not optimized to provide such. MEG and EBM_{0GP} estimate the density inaccurately in some boundary and peripheral areas. This may be caused by insufficient or excessive optimization of entropy as the zero-centered gradient penalty is not derived from the maximum likelihood principle. Both our $\text{EBM}_{\text{SV+GP}}$ and VERA successfully learn a sharp distribution, but our method is more stable in some inner regions of the 25-Gaussians and boundary regions of the swiss-roll. $\text{EBM}_{\text{MI+diff}}$ generates clearer details on 25 Gaussian but is less sharp on swiss-roll.

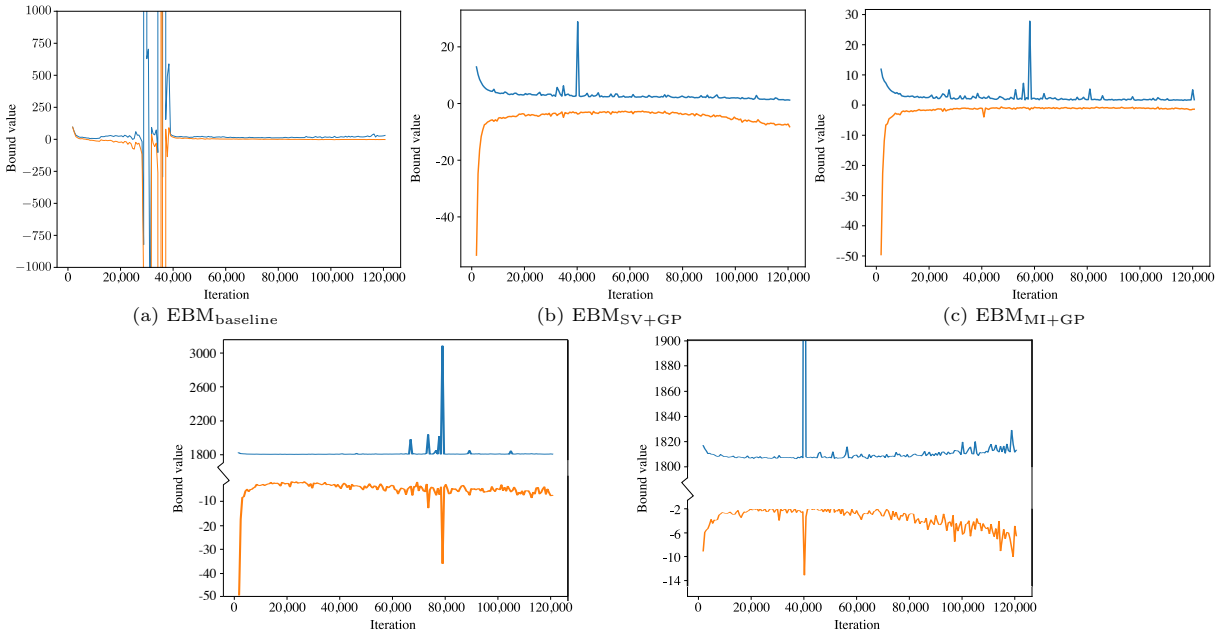
4.3 MNIST

Mode counting. Mode collapse is a common issue in generative models, where the generator G



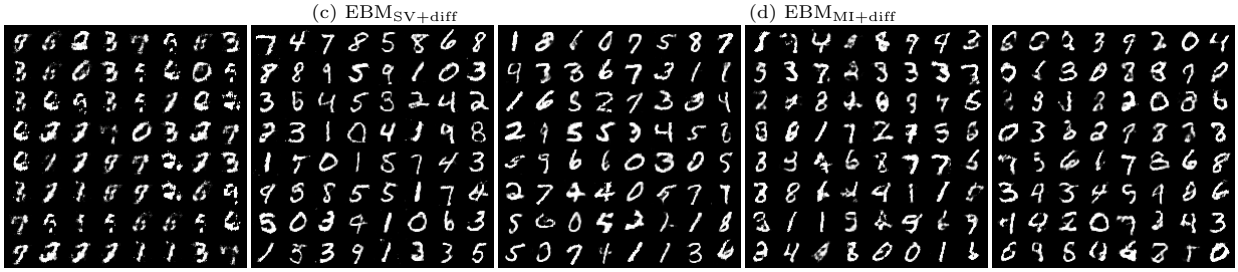
(a) Data (b) WGAN-0GP (c) MEG (d) VERA (e) EBM_{0GP} (f) $\text{EBM}_{\text{SV+GP}}$ (g) $\text{EBM}_{\text{MI+diff}}$

Fig. 4 Density estimation on the 25-Gaussians and swiss-roll datasets.



(a) $\text{EBM}_{\text{baseline}}$ (b) $\text{EBM}_{\text{SV+GP}}$ (c) $\text{EBM}_{\text{MI+GP}}$

(c) $\text{EBM}_{\text{SV+diff}}$ (d) $\text{EBM}_{\text{MI+diff}}$



(a) $\text{EBM}_{\text{baseline}}$ (b) $\text{EBM}_{\text{SV+GP}}$ (c) $\text{EBM}_{\text{MI+GP}}$ (d) $\text{EBM}_{\text{SV+diff}}$ (e) $\text{EBM}_{\text{MI+diff}}$

Fig. 5 Bidirectional bounds estimation for each combination of different bounds and their corresponding generation results.

maps all latent inputs to a reduced number of observation points. GANs, in particular, suffer from this issue. EBMs train the generator to have maximal entropy, and thus they often do well at capturing all modes of the data distribution. We follow the test procedure of Kumar et al. [44] to empirically verify that this is also the case for our EBMs. We train all models on the StackedMNIST dataset, which is a synthetically created dataset that stacks MNIST on different channels. The ground truth total number of modes is 1,000, which can empirically be counted using a pre-trained MNIST classifier. We

Table 1 Number of captured modes and KL divergence between the real and generated distributions.

Model	Modes↑	KL↓
DCGAN	392 ± 7.4	8.012 ± 0.056
SNGAN	441 ± 39.0	2.755 ± 0.033
WGAN-0GP	**1000 ± 0.0**	0.048 ± 0.003
MEG	**1000 ± 0.0**	0.042 ± 0.004
VERA	989 ± 9.0	0.152 ± 0.037
EBM_{0GP}	**1000 ± 0.0**	**0.039 ± 0.003**
$\text{EBM}_{\text{SV+GP}}$ (FC)	**1000 ± 0.0**	0.045 ± 0.003
$\text{EBM}_{\text{SV+GP}}$ (conv)	-	-
$\text{EBM}_{\text{MI+diff}}$ (FC)	988±2.36	0.401±0.040
$\text{EBM}_{\text{MI+diff}}$ (conv)	**1000 ± 0.0**	0.045 ± 0.003

Table 2 MNIST test log-likelihood on energy function with upper and lower bounds combinations

	MI	SV
0GP	-1819 ± 35.8	-2167 ± 507.2
GP	$\mathbf{-1782 \pm 132.6}$	-1634 ± 76.0
diff	-1798 ± 281.8	$\mathbf{-1561 \pm 297.5}$

further calculate the KL divergence between the generated mode distribution and the data distribution, and present both results in Table 1. As one would expect, GAN-based methods generally suffer from mode collapse with the exception of WGAN-0GP. The EBMs generally capture (nearly) all modes. Our EBM_{0GP} finds all modes and achieves the smallest KL divergence. It is worth emphasizing that WGAN-0GP obtains comparable results, which aligns with observations by Grathwohl et al. [20]. We also observe that our $\text{EBM}_{\text{SV+GP}}$ is sensitive to the choice of network structure, as it only gives good performance with a fully connected network, while it suffers from mode-collapse when using a CNN network. $\text{EBM}_{\text{MI+diff}}$, on the other hand, gives consistently good results across network architectures.

Entropy estimation. Next, we evaluate how well our entropy estimator based on singular values performs when measured on MNIST (Fig. 6). The lower bound derived from mutual information uses the Jensen-Shannon divergence rather than the KL divergence, which hinders direct entropy estimation. Hence we do not include this in the comparison. We here consistently use networks with fully-connected layers as we can then easily derive the Jacobian in closed form, which provides us with the ground truth entropy of the generator. We compare our estimator (red) with the ground truth

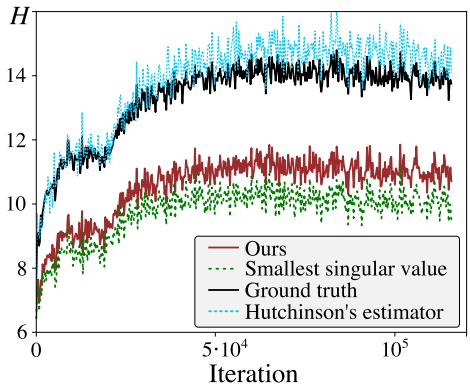


Fig. 6 Entropy estimators with Hutchinson's estimator, true smallest singular value based estimator, and our approximation in singular value lower bound.



(a) $\text{EBM}_{\text{SV+GP}}$



(b) $\text{EBM}_{\text{MI+diff}}$

Fig. 7 The negative energy of images generated by linear interpolation of two real images. Real images have higher negative energy than synthetic ones.

(black) and the variant of Hutchinson's estimator (blue) proposed by Kumar et al. [43]. Finally, to investigate the choice of only taking a few steps in our calculation of the singular values, we also compare with a high-precision estimator of the smallest singular value (green). We see that the variant of Hutchinson's estimator is quite close to the ground truth, but that it consistently upper bound the ground truth such that it is not suitable to use as a lower bound. Compared to the high-precision singular value estimator, we see a noticeable difference to our low-precision estimator. It is worth noting that by definition our low-precision estimator upper bounds the high-precision one. Finally, we observe that all estimators follow similar trends, indicating that our estimator is suitable for optimization.

Test log-likelihood. To compare different combinations of bounds, we consider an example where we can evaluate the exact log-likelihood associated with our energy function. To ensure that we can evaluate the exact log-likelihood, we use a normalizing flow [13] for the energy function. In this case, for $\text{EBM}_{\text{SV+GP}}$,

$E_\theta(\mathbf{x}) = -\log p_\theta(\mathbf{x}) = -\log p_z(f_\theta(\mathbf{x})) - \log\left(\left|\det\left(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right|\right)$, where $f_\theta(\mathbf{x})$ is a invertible neural network. Likewise, for $\text{EBM}_{\text{MI+diff}}$, $E_\theta(\mathbf{x}, t) = -\log p_{\theta t}(\mathbf{x}) = -\log p_z(f_\theta(\mathbf{x}, t)) - \log\left(\left|\det\left(\frac{\partial f_\theta(\mathbf{x}, t)}{\partial \mathbf{x}}\right)\right|\right)$. As discussed in Sec. 2.4, this need not be a valid bound as it may not satisfy the Fokker-Planck equation, but empirically we found it to be stable. We train six different models by considering all combinations of our proposed bounds as well as a zero-centered gradient penalty (0GP) as an approximate upper bound. The latter is included in the comparison as it is widely recognized as a common trick for stabilizing training, even though it does not constitute a bound. However, it exhibits instability in our experiment. Table 2 shows test set log-likelihoods for the different bound combinations for training. We observe that except for the 0GP-based upper bound, both the gradient penalty and score-based diffusion upper bound give satisfying results with both lower bounds. Among them, the singular value lower bound performs slightly better than the mutual information lower bound with both upper bounds. In fact, the heuristically defined zero-centered gradient penalty demonstrates instability during training when used with the singular value lower bound. Thus we set the number of inner loop iterations in SV to zero, which means a Hutchinson estimator for computing the entropy.

Bidirectional bounds. To verify the bidirectional characteristic of our method, we plot the upper and lower bound during training for each combination. As before, we use a normalizing flow model as an energy function. Figure 5 show the results. Note that the diffusion-based upper bound depends on a constant that we cannot evaluate (28), which may explain why the upper bound in $\text{EBM}_{\text{MI+diff}}$ and $\text{EBM}_{\text{SV+diff}}$ are quite loose. We do not have a tightness measure for this upper bound. The gradient penalty upper bound equals the lower bound when $p_\theta(\mathbf{x}) = p_g(\mathbf{x})$. Since we approximate the entropy used for computing the gradient penalties upper bound, we also cannot give an exact tightness measure here. We can, however, give the training trends for each combination of bidirectional bounds. In Fig. 5, we observe that, except for the baseline model without bidirectional bounds, all combinations of bidirectional bounds can give stable training curves, yielding decent generation results. No notable visualization differences in the generation performance between each combination.

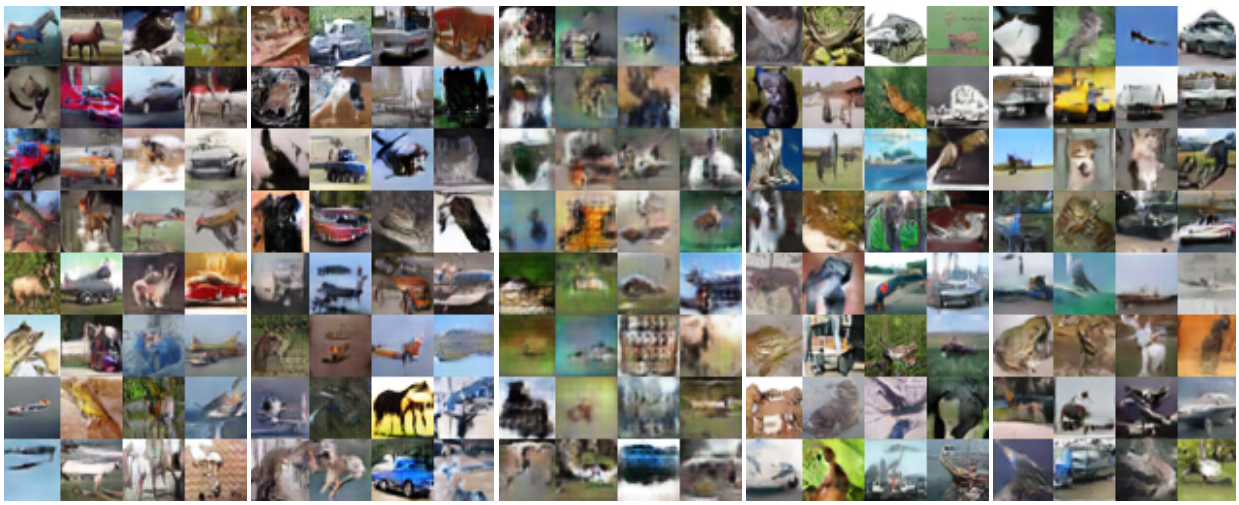
Energy estimation. To further verify if our energy can provide useful information about the density of data space, we linearly interpolate two randomly chosen images and compute the negative energy of the interpolated images. We use CNNs for both energy function E_θ, generator G, and discriminator T. From Fig. 7 we observe that the real images have a higher likelihood under the energy functions than the synthetic (interpolated) images. This suggests that adversarial training pushes probability mass toward real images.

4.4 Natural images

Image generation. So far we have only considered toy data, and do not challenge contemporary generative models, including our EBMs. Next we, therefore, consider more challenging benchmark data, specifically the 32×32 CIFAR-10 [42] dataset, the 64×64 cropped ANIMEFACE[4] dataset, and the 64×64 CelebA [48] dataset. All represent a significant increase in complexity. For CIFAR-10 and ANIMEFACE datasets, we report Inception Score (IS), Fréchet Inception Distance (FID) score, and two F_β scores [59] to measure model performance. Table 4 provides results for both our methods and the considered baselines, where we note that all adversarial-based EBMs and GANs use the same network architecture. We use the DCGAN [57] network architecture for CIFAR-10 and a Resnet architecture [38] for ANIMEFACE. Hyper-parameters are chosen as in the original papers. We were unable to reproduce the reported results for VERA [20] so we instead performed an extensive grid search to select the best hyperparameters. Like the original paper, we choose the entropy weight to be 0.0001. We relied on a public implementation[5] for DDPM. For EBM-Triangle, we reproduce results using the official code. For other models, we take the results from the original papers.

Table 4 shows that among the EBMs, our $\text{EBM}_{\text{SV+GP}}$ generally performs the best on CIFAR-10, though $\text{EBM}_{\text{MI+diff}}$ has the best IS score. MEG, EGAN-Ent-VI, and VERA derive mutual

[4] https://www.kaggle.com/splcher/animefacedataset
[5] https://github.com/rosinality/denoising-diffusion-pytorch



(a) WGAN-0GP (b) MEG (c) VERA (d) $\text{EBM}_{\text{SV+GP}}$ (e) $\text{EBM}_{\text{MI+diff}}$

Fig. 8 Generated samples on CIFAR-10 with our method and various methods.



(a) WGAN-0GP (b) MEG (c) VERA (d) $\text{EBM}_{\text{SV+GP}}$ (e) $\text{EBM}_{\text{MI+diff}}$

Fig. 9 Generated samples on ANIMEFACE with our method and other generative models

information or variational approximations for the intractable entropy terms. EBM-Triangle and EBM-VAE use the joint distribution between latent and data space to avoid the intractable entropy term. But they all minimize a lower bound when optimizing the energy function. The EBMs are, however, generally outperformed by NCSN and DDPM, which is not surprising as NCSN and DDPM both use UNet architecture which is much bigger and more complex than our simple DCGAN architecture. To demonstrate this, we also scale up our network to UNet and get comparable results with NCSN and DDPM, see Sec. D.1 for details.

On ANIMEFACE we see that our method is highly competitive. Figures 8 and 9 show samples from different models. On ANIMEFACE, our EBMs yield more diverse samples with respect to face parts than the competing methods. All models mainly generate feminine faces. This could indicate that all models to some extent suffer from mode collapse, though our methods seem less sensitive to this issue. We further draw attention to the many corrupt samples drawn from VERA, despite extensive parameter search. In contrast, with CIFAR-10, we observed no notable differences in the generative capabilities between models.

For CelebA dataset, we report the FID score in Table 5 since this measure metric is the most common score for comparison of current generative models. We use 162,770 images for training and 19,867 for testing, the official split in PyTorch. The FID score is computed with testing images as the reference images. From Table 5 we can see our $\text{EBM}_{\text{SV+GP}}$ get the best results among GANs and EBMs except for EBM-Diffusion. DDPM and EBM-Diffusion get the best FID score. Both methods use a diffusion process to learn data distribution, which can obtain the advantage of a diffusion-based model on generation quality. But they also have the disadvantage of being computationally expensive. Our $\text{EBM}_{\text{MI+diff}}$ performs slightly worse on CelebA but still gets comparable results with other EBMs.

Capacity usage. In general, we want the proposal distribution p_g to use as much of the capacity of the generator network G as possible in order to be adaptive. To compare across models, we consider the anisotropy index [75], which is an implicit measure of capacity usage $\mathcal{C}_{\mathbf{z}}$ that locally measures the intrinsic dimensionality of $G(\mathbf{z})$. Specifically, for a given $\mathbf{z}$, this measures the standard deviation of the norm of the directional derivative of $G(\mathbf{z})$ with respect to the input dimensions, i.e.

$$\mathcal{C}_{\mathbf{z}} = \text{std}\left(\{\|\mathbf{J}_{\mathbf{z}}\mathbf{e}_i\|\}_{i=1\ldots d}\right), \tag{34}$$

where $\text{std}(\cdot)$ denotes the standard deviation, and $\mathbf{e}_i$ is the i^{th} standard basis vector. Small values of $\mathcal{C}_{\mathbf{z}}$ are indicative that all input dimensions contribute equally to the output of G, which suggests good capacity usage. We report the average value of $\mathcal{C}_{\mathbf{z}}$ for $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ across several runs in Table 3. Most models, except VERA, perform well on CIFAR-10, while performance on ANIMEFACE is more diverse. In both cases, $\text{EBM}_{\text{SV+GP}}$ and $\text{EBM}_{\text{MI+diff}}$ show the most capacity usage. The significant difference between EBM_{0GP} and the bidirectional EBMs ($\text{EBM}_{\text{SV+GP}}$ and $\text{EBM}_{\text{MI+diff}}$) on ANIMEFACE suggests that the use of an upper bound helps to maximize the entropy of the proposal distribution p_g.

Out-of-distribution detection. In principle, a well-estimated EBM should assign a lower likelihood to out-of-distribution (OOD) observations than to observations from the training distribution. OOD detection performance based on likelihood can then serve as a measure of density quality. We compare the relevant baselines following established practice [4, 9, 26–28, 58] and report the threshold independent evaluation metrics of Area Under the Receiver Operator Characteristic (AUROC↑), Area Under the Precision-Recall Curve (AUPRC↑) and False Positive Rate at 80% (FPR80↓). Here the arrow indicates the direction of improvement of the metrics. Table 6 provide the results, where each column takes the form 'In-distribution / Out-of-distribution' in reference to

Table 3 Anisotropy indices (↓).

Model	CIFAR-10	ANIMEFACE
WGAN-0GP	1.0823 ± 0.005	2.4366 ± 0.129
MEG	0.9467 ± 0.009	2.3600 ± 0.142
VERA	3.9694 ± 0.055	2.2929 ± 0.2074
EBM_{0GP}	1.0688 ± 0.0226	3.4560 ± 0.0521
$\text{EBM}_{\text{SV+GP}}$	**0.9431 ± 0.0191**	**1.8016 ± 0.1345**
$\text{EBM}_{\text{MI+diff}}$	1.0092 ± 0.0107	1.9067 ± 0.0617

Table 4 Comparison in terms of FID, Inception Score, F_8 score (weights recall higher than precision) and $F_{1/8}$ score (weights precision higher than recall).

Model	Inception↑	FID↓	F_8↑	$F_{1/8}$↑
CIFAR-10				
DCGAN	6.64	49.03	0.795	0.83
WGAN-0GP	7.24 ± 0.035	29.31 ± 0.185	0.92 ± 0.010	0.95 ± 0.010
CoopNets [2018a]	6.55	36.4	-	-
CoopNets [2018b]	-	33.61	-	-
EGAN-Ent-VI [11]	7.07 ± 0.10	-	-	-
EBM-VAE [85]	6.65	36.2	-	-
EBM-Triangle [24]	7.30	28.96	0.89	0.96
IGEBM [14]	6.78	38.2	-	-
Short-Run MCMC [53]	6.21	44.50	-	-
MEG	6.62 ± 0.243	34.55 ± 1.145	0.88 ± 0.001	0.92 ± 0.010
VERA	5.06 ± 0.555	66.38 ± 6.635	0.58 ± 0.080	0.79 ± 0.005
NCSN	8.87 ± 0.12	25.32	-	-
DDPM	**9.03**	**7.76**	**0.98**	**0.99**
EBM_{0GP}	6.90 ± 0.032	35.42 ± 0.582	0.90 ± 0.004	0.93 ± 0.002
$\text{EBM}_{\text{SV+GP}}$	7.45 ± 0.014	28.63 ± 0.290	0.93 ± 0.001	0.95 ± 0.008
$\text{EBM}_{\text{MI+diff}}$	7.46±0.051	30.37 ± 0.054	0.92 ± 0.003	0.95 ± 0.004
ANIMEFACE				
WGAN-0GP	2.22 ± 0.030	9.76 ± 0.674	**0.95 ± 0.005**	**0.98 ± 0.005**
MEG	2.20 ± 0.020	9.31 ± 0.007	**0.95 ± 0.005**	**0.98 ± 0.001**
VERA	2.15 ± 0.001	41.00 ± 1.072	0.515 ± 0.078	0.78 ± 0.013
DDPM	2.18	**8.81**	0.94	**0.98**
EBM_{0GP}	**2.26 ± 0.017**	20.53 ± 0.524	0.889 ± 0.008	0.909 ± 0.019
$\text{EBM}_{\text{SV+GP}}$	**2.26 ± 0.005**	12.75 ± 0.045	0.94 ± 0.001	0.96 ± 0.005
$\text{EBM}_{\text{MI+diff}}$	2.17±0.065	13.32±0.45	0.92±0.001	0.97±0.004

Table 5 FID scores on 64 × 64 CelebA dataset

Model	FID
WGAN-0GP	8.59 ± 0.179
MEG	8.52 ± 0.042
Joint-EBM-Triangle [25]	24.7
BiDVL [37]	17.24
EBM-Diffusion [16]	5.98
DDPM	**3.50**
EBM_{0GP}	10.72 ± 0.314
$\text{EBM}_{\text{SV+GP}}$	8.39 ± 0.567
$\text{EBM}_{\text{MI+diff}}$	14.79±1.08

the training and test set, respectively. On CIFAR-10 / SVHN we note that $\text{EBM}_{\text{SV+GP}}$ works the best. On CIFAR-10 / CIFAR-100 we generally observed poorer results, but EBM_{0GP} did quite a bit better than the rest. The general degradation is most likely due to the strong similarity between CIFAR-10 and CIFAR-100. For ANIME-FACE / Bedroom, however, a similar situation occurs, even if the datasets are not alike. Here $\text{EBM}_{\text{MI+diff}}$ is significantly better than the rest.

This study provides no clear winner. Our approach consistently performs well, but perhaps surprisingly, this is also the case for WGAN-0GP, which is not a density estimator.OOD detection is a task known for its subtle pitfalls [26] and high variance across training stages in EBMs [37], so we argue that the presented results should be taken with a grain of salt even if our model performs well.

Comparison of different combinations of bounds. To explore the performance of different combinations of upper and lower bounds on natural images, we do a comparison with different bounds on CIFAR-10 across different metrics. Table 7 shows the results. We do not see a clear winner in this comparison. For the computational cost, $EBM_{\text{MI+diff}}$ is time efficient and is around $1.3\times$ of WGAN-0GP, other types of our bidirectional EBMs require to compute the smallest singular value and are around $2.5\times$ of baseline. $\text{EBM}_{\text{SV+GP}}$ and $\text{EBM}_{\text{SV+diff}}$ have no extra network and are space-efficient. Hence, the selection of bidirectional bounds depends on training requirements, such as finding the optimal balance between memory usage and training time.

Robustness of upper bound. We next test the robustness of our two upper bounds to changes in parameters. As described in Sec. 4.1, when using the singular value-based lower bound on large-scale datasets we reduce the weight of entropy regularizer λ to improve performance. We combine this lower bound with our two upper bounds respectively and consider different values of λ. Results on CIFAR-10 are in Table 8. We see that with the diffusion-based bound we always get satisfying performance, while the gradient penalty bound results in a loss in performance when $\lambda = 1$.

5 Discussion

This paper stated from the observation that current GAN-like methods for training energy-based models (EBMs) both minimize and maximize a lower bound. We have put forward the hypothesis that this may cause training instability, and have instead proposed to bound the negative log-likelihood from above and below and switch between bounds when minimizing and maximizing. We then propose a series of such bounds and compare their empirical behavior. The lower bound based on singular values is similar to existing ones, but the proposed algorithm for its evaluation is new. Unlike previous methods, this new algorithm does not require any additional networks and is generally easy to tune; the only parameter is the number of iterations, which reflects a trade-off between computational cost and the accuracy of the bound. The mutual information-based lower bound needs an extra network but it is, in turn, time efficient. The two lower bounds then respectively give space or time efficiency.

The idea of introducing an upper bound, $\lceil L(\theta) \rceil$, is new to the literature. Our gradient penalty-based bound is, however, similar to commonly called upon heuristics to stabilize training. To the best of our knowledge, this is the first time that gradient penalties have been shown to provide a bound of the log-likelihood. We consider it rather neat that current engineering practice, thus, can be justified directly from a probabilistic perspective.

The general empirical finding across datasets and tasks is that our bidirectional bounds generally lead to models that perform as well or better than some of the current state-of-the-art. The evidence suggests that the use of bidirectional bounds causes the generator to increase both its entropy and its capacity usage. We observe this both through direct measurement and improved sample quality.

Limitations. We mainly have four concerns with the proposed methods. First, for our score-based diffusion upper bound, it is not clear how to design $E_\theta(\mathbf{x}, t)$ to make it satisfy the Fokker-Planck equation, and this part remains heuristic. Second, our lower bound based on singular values relies on an iterative solver. While our implementation is quite efficient, it must still run for several iterations to ensure convergence. Our practical implementation, therefore, stops prematurely to ensure fast computations. We found this to empirically work well even if it technically violates the bound. Furthermore, to get good performance we need a weight regularizer λ to balance the entropy

Table 6 AUROC↑, AUPRC↑, and FPR80↓ for OOD detection for 'train/test' datasets.

Model	CIFAR-10 / SVHN			CIFAR-10 / CIFAR-100			ANIMEFACE / Bedroom		
	AUROC↑	AUPRC↑	FPR80↓	AUROC↑	AUPRC↑	FPR80↓	AUROC↑	AUPRC↑	FPR80↓
WGAN-0GP	0.8	0.83	0.39	0.54	0.55	0.77	0.51	0.48	0.72
MEG	0.79	0.81	0.42	0.52	0.53	0.8	0.56	0.53	0.77
VERA	0.62	0.64	0.64	0.51	0.51	0.79	0.60	0.525	0.6
EBM_{0GP} (ours)	0.66	0.69	0.67	**0.64**	**0.63**	**0.64**	0.67	0.63	0.53
EBM_{SV+GP} (ours)	**0.88**	**0.86**	**0.1997**	0.53	0.52	0.7765	0.53	0.505	0.83
$EBM_{MI+diff}$ (ours)	0.53	0.52	0.73	0.52	0.51	0.78	**0.84**	**0.72**	**0.201**

Table 7 Comparison with combinations of different upper and lower bounds on the CIFAR-10 dataset.

Metric	GP		diff	
	MI	SV	MI	SV
FID↓	28.96±1.251	**28.63 ± 0.290**	30.37±0.054	29.31±0.261
IS↑	7.41±0.093	7.45 ± 0.014	7.46±0.051	**7.65±0.008**
F8↑	0.92±0.004	**0.93 ± 0.001**	0.92±0.003	0.93±0.004
F1/8↑	**0.96±0.003**	0.95 ± 0.008	0.95±0.004	0.95±0.005

term in the lower bound based on singular values. In general, this weight regularizer violates the lower bound, but in practice, we almost always have positive entropies, in which case the bound remains valid. Finally, for the gradient penalty upper bound, we need access to the volume of support M for the proposal distribution p_g. We do not have a method to compute this and therefore consider the ratio $\frac{M}{s_1^2}$ between M and the square of the estimated smallest singular value s_1^2 as a parameter that is either fixed or dynamic decay. When used in conjunction with the singular value lower bound, we observe that $\frac{M}{s_1^2}$ needs to be adjusted with the weight regularizer λ to get a satisfying performance.

Declarations

5.1 Funding

This work was supported by the National Natural Science Foundation of China under Grant U24A20220 and partly funded by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606), STCSM under Grant 22DZ2229005, and 111 project BP0719010. It also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research, innovation program (757360), National science foundation of China under grant 61771305, and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). JF was supported in part by the Novo Nordisk Foundation (NNF20OC0065611) and the Independent Research Fund Denmark (9131-00082B). SH was supported in part by research grants (15334, 42062) from VILLUM FONDEN. The authors also acknowledge the support of the Pioneer Centre for AI, DNRF grant number P1.

5.2 Competing interests

- The authors have no relevant financial or non-financial interests to disclose.
- The authors have no competing interests to declare that are relevant to the content of this article.
- All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.
- The authors have no financial or proprietary interests in any material discussed in this article.

5.3 Data availability

All data analyzed during this study are publicly accessible.

Appendix A Model architecture

Our key motivation for design of p_g is to ensure that the model has support over the entire data space. This is needed for the reason that the KL divergence requires the two distributions to be defined on the same measurable space. Since our energy distribution is defined for the entire data space, we need the generated distribution to have

Table 8 Robustness of upper bounds with different entropy regularizer weight λ.

	diff			GP		
λ	0.0001	0.01	1	0.0001	0.01	1
FID↓	29.31±0.261	30.4±0.65	33.13±0.349	28.63±0.290	30.38±0.22	45.08±0.285
IS↑	7.65±0.008	7.35±0.079	7.12±0.079	7.45±0.014	7.17±0.085	6.16±0.126
F8↑	0.931±0.004	0.906±0.006	0.903±0.002	0.93±0.001	0.91±0.002	0.78±0.007
F1/8↑	0.95±0.005	0.956±0.000	0.94±0.004	0.95±0.008	0.95±0.002	0.91±0.006

the same support. The simplest way to achieve this is with our choice of $x = G(z) + \epsilon$, where ϵ is a small-variance Gaussian variable. This concentrates the distribution around the manifold spanned by our generator while ensuring that the generated distribution has full support over the data space. Moreover, if we do not add noise to our generator, our mutual information lower bound is infinite. The added noise makes the bound finite.

We consistently assume that $G : \mathbb{R}^d \to \mathbb{R}^D$ spans an immersed d-dimensional manifold in $\mathbb{R}^D$ as this allows us to apply the change-of-variables formula to get a density for the generator. In order to ensure that this assumption is valid, we minimally restrict the architecture of the generator neural network.

An immersion implies that the Jacobian of G exists and has full rank. We can then ensure existence by using activation functions with at least one derivative almost everywhere. Any smooth activation satisfies this assumption, but we emphasize that, for example, ReLU activations lack a derivative at a single point. The non-smooth region has measure zero if the linear map inside the activation is not degenerate, in which case the change-of-variables technique still applies.

We are unaware of architectural tricks to ensure that the Jacobian has full rank. Still, a minimum requirement is no hidden layer may have dimensionality less than the d dimensions of the latent space. This is a natural requirement that is practically always satisfied. Our model maximizes the generator entropy, which practically ensures full-rank Jacobians (noting that degeneracy in the Jacobian reduces the entropy). We have in practice generally not experienced degenerate Jacobians during training.

Appendix B Training details

Table A1 gives the hyper-parameters used during training. In Eq. (33) we set $p = 2$. We normalize the data to $[-1, 1]$ and do not use dequantization. We augment only using random horizontal flips during training. We used the official train-test split from PyTorch for cifar10 and an 85:15 train-test split for animeface. We set $\frac{M}{s_1^2} = \frac{0.1}{z_{dim}}$ for our gradient penalty upper bound, where s_1 is the smallest singular value of Jacobian and z_{dim} is the latent dimension. We find this set of parameters generally works well across datasets. Furthermore, we choose $\frac{M}{s_1^2}$ with dynamic decay, i.e. $\frac{M}{s_1^2} = \frac{a(b+i)^{-\gamma}}{2}$ where i denotes the iteration, $\gamma = .55$ and a and b are chosen such that $\frac{2M}{s_1^2}$ decreases from 0.01 to 0.0001 during training. These dynamics are consistent with the approximation of $\mathrm{KL}\left(p_g \| p_\theta\right)$ from the Taylor series point of view. We find that practically for $\mathrm{EBM}_{\mathrm{SV+GP}}$, $\frac{M}{s_1^2}$ must be adjusted with entropy regularizer λ together. We also observe the implications of the network design on the final performance. For example, not using batch normalization in the energy function can improve the stability of training on ANIMEFACE dataset.

Appendix C Selection of bidirectional bound pairings

Given the two available options for both the lower and the upper bound, we provide recommendations for selecting bidirectional bound pairings based on our training experience. For real-world applications, as elaborated in the main text, we generally recommend using combinations $\mathrm{EBM}_{\mathrm{SV+GP}}$ or $\mathrm{EBM}_{\mathrm{MI+diff}}$ first due to their respective advantages in memory and time efficiency. If the data and latent space dimensions are relatively small

Table A1 Selection of most important hyper-parameters and their setting.

Datasets	Optimization	Learning rate	Batch size	Iterations/Epochs	latent dim
Toy	Adam(0.0,0.9)	2e-4	200	150000	2
MNIST	Adam(0.0,0.9)	2e-4	100	90(epochs)	32
CIFAR-10	Adam(0.0,0.999)	5e-5	64	200000	128
ANIMEFACE	Adam(0.0,0.999)	5e-5	64	100000	128
CelebA	Adam(0.0,0.999)	5e-5	64	200000	128

and the network is shallow, we recommend using $\text{EBM}_{\text{SV+GP}}$, as it tends to achieve better performance in such scenarios. Conversely, for larger dimensions or deeper networks, we suggest using $\text{EBM}_{\text{MI+diff}}$ to reduce computational cost. Furthermore, GP and SV may suffer from training instability due to their reliance on iterative optimization. Therefore, in cases of unstable training, it is advisable to consider replacing GP or SV with diff or MI.

Appendix D Additional results

D.1 CIFAR-10

Although our bidirectional EBM obtains superior performance among EBMs, it still lags far behind diffusion-based models in terms of generation metrics. To demonstrate the scalability and potential of our model, we also scale up our network structure to UNet for CIFAR-10 dataset. We adopt the same architecture in UOTM [10], where its discriminator is utilized as our energy function. Our generator has the same scale as the network in NCSN and DDPM. We borrow some training strategies from UOTM to stabilize training and improve generative performance, such as the form of energy discrepancy and cost function. For the choice of bidirectional bounds, we only use our $\text{EBM}_{\text{MI+diff}}$ since UNet network is complex and the input has the same dimension as the output, causing a costly Jacobian computation. We present our quantitative results in Table D2 and qualitative results in Fig. D2. All baseline results used for comparison are sourced from Xiao et al. [77] and Song et al. [71]. For our method, we run the experiments with different seeds and report the standard deviation. For FID evaluation, we use the training set as a reference, aligning common practice in diffusion-based models [77]. From Table D2 we can observe our $\text{EBM}_{\text{MI+diff}}$ achieves results comparable to diffusion-based models while using only one-step generation, offering significantly greater efficiency. Although recent techniques like distillation and consistency models can also enable one-step generation with minimal performance decline, our model demonstrates competitive results compared with them. This experiment suggests that the superiority of diffusion-based models stems more from the large network architecture and meticulous parameter fine-tuning than from the methodology itself. Our bidirectional EBM can achieve performance comparable to recent advanced diffusion-based models when equipped with a similar-scale UNet and an appropriate training strategy. Moreover, as shown in [18], EBM can be further improved by integrating it with a diffusion process, which presents an opportunity for future work.

Table D2 Comparison with diffusion-based models on 32×32 CIFAR-10 dataset.

Model	FID↓	IS↑	NFE↓
DDPM [33]	3.21	9.46	1000
NCSN [67]	25.3	8.87	1000
DDIM [66]	4.67	8.78	50
Score SDE (VE) [70]	**2.20**	9.89	2000
Score SDE (VP) [70]	2.41	9.68	2000
DDPM Distillation [49]	9.36	8.36	1
CD [71]	3.55	9.48	1
CT [71]	8.70	8.49	1
$\text{EBM}_{\text{MI+diff}}$ (Large)	5.47 ± 0.14	**10.20±0.09**	1

D.2 Imagenet

To further evaluate the efficacy of our model on complex datasets that better approximate real-world scenarios, we assess its generative performance on 32×32 ImageNet dataset, which contains 1,281,167 training images and 50,000 test images.

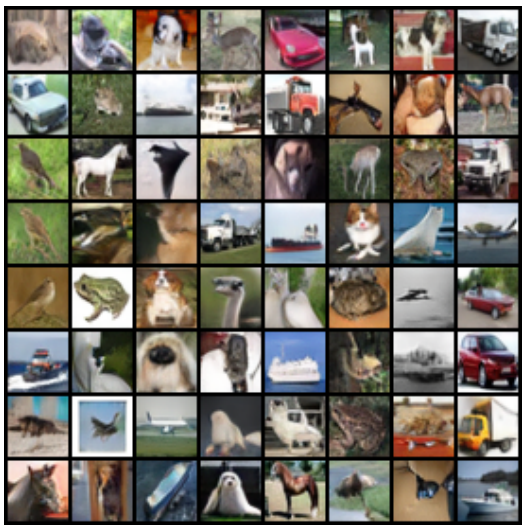


Fig. D1 Visual results of CIFAR-10 using UNET.

Table D3 FID scores on 32 × 32 ImageNet dataset

Model	FID
EBM-IG [14]	62.23
PixelCNN [62]	40.51
CLEL-Large [46]	15.47
DDPM++(VP, NLL) [39]	8.42
CDRL [89]	9.35
$\text{EBM}_{\text{SV+GP}}$	11.46 ± 0.236
$\text{EBM}_{\text{MI+diff}}$	14.61 ± 0.371
$\text{EBM}_{\text{MI+diff}}$ (Large)	**6.57 ± 0.114**

We perform experiments using both DCGAN and UNet architectures, as applied to the CIFAR-10 dataset. For $\text{EBM}_{\text{SV+GP}}$, we set the number of inner loop iterations in SV to zero and find that they contribute to stabilizing the training process; a detailed exploration of this phenomenon is left for future work. Table D3 and Fig. D3 show our quantitative and qualitative results respectively. For quantitative evaluation, we report only the FID score, as it is the most widely used metric for assessing generative models on this dataset. From Table D3 we can see our $\text{EBM}_{\text{MI+diff}}$-Large gets the best performance compared to other advanced generative models including diffusion-based model DDPM++ [39] and the latest energy-based model CDRL [89]. Even our $\text{EBM}_{\text{SV+GP}}$ and $\text{EBM}_{\text{MI+diff}}$ only use simple DCGAN architecture, they still outperform competitive EBM framework CLEL-Large which employs a deep ResNet network. This experiment further verifies the effectiveness of our bidirectional EBM on large complex datasets.

D.3 LSUN Church

We also test our model on a large-scale LSUN dataset. For this dataset, we choose the Church outdoor category with a size of 128 × 128. We use 126,227 training images and 300 testing images following the official PyTorch split. The latent dimension is set to 256. We employ the same ResNet architecture as that used for the ANIMEFACE and CelebA datasets. We trained $\text{EBM}_{\text{SV+GP}}$ for 100,000 iterations and $\text{EBM}_{\text{MI+diff}}$ for 200,000 iterations. Fig. D2 illustrates the scalability of our models. While these samples are not state-of-the-art, they do showcase the ability of our model to scale to high-dimensional problems. Further improving these results most likely requires work on picking the right network architecture, fine-tuning parameters, and specialized optimization strategies. We leave this for future work.



(a) $\text{EBM}_{\text{SV+GP}}$



(b) $\text{EBM}_{\text{MI+diff}}$

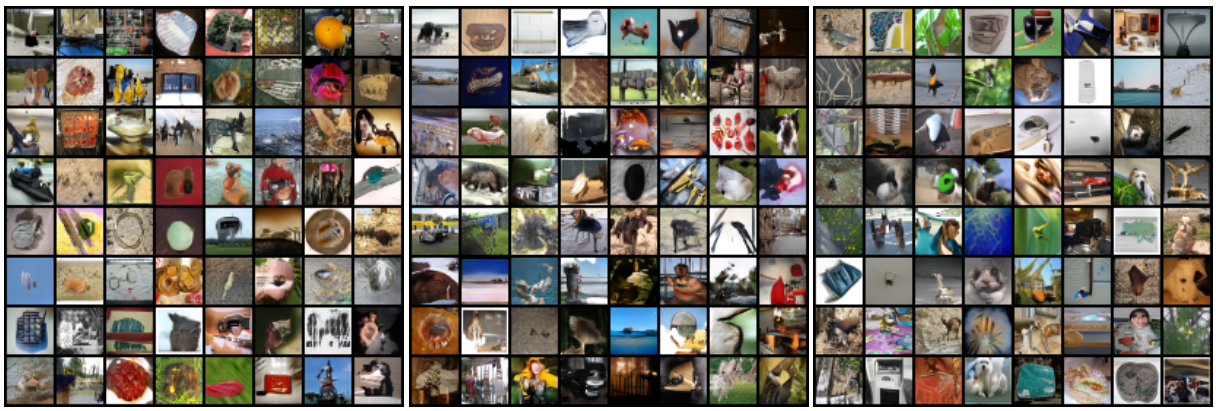
Fig. D2 Visual results of 128×128 LSUN Church with two different settings.

References

[1] M Ehsan Abbasnejad, Qinfeng Shi, Anton van den Hengel, and Lingqiao Liu. A generative adversarial density estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10782–10791, 2019.

[2] M Ehsan Abbasnejad, Javen Shi, Anton van den Hengel, and Lingqiao Liu. Gade: A generative adversarial approach to density estimation and its applications. *International Journal of Computer Vision*, 128(10): 2731–2743, 2020.

[3] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm



(a) $\text{EBM}_{\text{SV+GP}}$ (b) $\text{EBM}_{\text{MI+GP}}$ (c) $\text{EBM}_{\text{MI+diff}}$ (Large)

Fig. D3 Visual results of 32×32 ImageNet dataset with different settings.

for boltzmann machines. *Cognitive Science*, 9 (1):147–169, 1985.

[4] Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. In *Uncertainty in Artificial Intelligence Workshop*, 2018.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[6] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[7] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Accurate and conservative estimates of MRF log-likelihood using reverse annealing. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38, pages 102–110. PMLR, 09–12 May 2015.

[8] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems*, volume 33, pages 12275–12287, 2020.

[9] Hyunsun Choi, Eric Jang, and Alexander A Alemi. WAIC, but why? Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[10] Jaemoo Choi, Jaewoong Choi, and Myungjoo Kang. Generative modeling through the semi-dual formulation of unbalanced optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[12] Adji B Dieng, Francisco JR Ruiz, David M Blei, and Michalis K Titsias. Prescribed generative adversarial networks. *arXiv preprint arXiv:1910.04302*, 2019.

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations Workshop*, 2015.

[14] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[15] Jes Frellsen, Ole Winther, Zoubin Ghahramani, and Jesper Ferkinghoff-Borg. Bayesian generalised ensemble markov chain monte

carlo. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 408–416. PMLR, 09–11 May 2016.

[16] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *International Conference on Learning Representations*, 2021.

[17] Cong Geng, Jia Wang, Zhiyong Gao, Jes Frellsen, and Søren Hauberg. Bounds all around: training energy-based models with bidirectional bounds. *Advances in Neural Information Processing Systems*, 34:19808–19821, 2021.

[18] Cong Geng, Tian Han, Peng-Tao Jiang, Hao Zhang, Jinwei Chen, Søren Hauberg, and Bo Li. Improving adversarial energy-based model via diffusion process. *Proceedings of the 41th International Conference on Machine Learning*, 2024.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[20] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations*, 2021.

[21] Roger B Grosse, Chris J Maddison, and Russ R Salakhutdinov. Annealing between distributions by averaging moments. *Advances in Neural Information Processing Systems*, 26, 2013.

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[23] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[24] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8670–8679, 2019.

[25] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Joint training of variational auto-encoder and latent energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7978–7987, 2020.

[26] Jakob D Havtorn, Jes Frellsen, Søren Hauberg, and Lars Maaløe. Hierarchical vaes know what they don't know. In *International Conference on Machine Learning*, pages 4117–4128. PMLR, 2021.

[27] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

[28] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019.

[29] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[30] Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 448, 1983.

[31] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep

belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[32] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[33] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[34] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424.

[35] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3): 1059–1076, 1989.

[36] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6 (4), 2005.

[37] Ge Kan, Jinhu Lü, Tian Wang, Baochang Zhang, Aichun Zhu, Lei Huang, Guodong Guo, and Hichem Snoussi. Bi-level doubly variational learning for energy-based latent variable models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18460–18469, 2022.

[38] Minguk Kang and Jaesik Park. ContraGAN: Contrastive learning for conditional image generation. In *Advances in Neural Information Processing Systems*, volume 33, pages 21357–21369, 2020.

[39] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[40] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[41] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2): 517–541, 2001.

[42] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[43] Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized autoencoders via relaxed injective probability flow. In *International Conference on Artificial Intelligence and Statistics*, pages 4292–4301. PMLR, 2020.

[44] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

[45] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[46] Hankook Lee, Jongheon Jeong, Sejun Park, and Jinwoo Shin. Guiding energy-based models via contrastive latent variables. In *International Conference on Learning Representations*, 2023.

[47] Yingzhen Li and Richard E. Turner. Gradient estimators for implicit models. In *International Conference on Learning Representations*, 2018.

[48] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[49] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

[50] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[51] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[52] Radford M Neal et al. MCMC using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[53] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

[54] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5272–5280, 2020.

[55] Takayuki Osogami. Boltzmann machines and energy-based models. *arXiv preprint arXiv:1708.06008*, 2017.

[56] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 Workshop Autodiff*, 2017.

[57] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[58] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[59] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

[60] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455. PMLR, 16–18 Apr 2009.

[61] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 872–879, 2008.

[62] Tim Salimans, Andrej Karpathy, Xi Chen, and DiederikP. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, Jan 2017.

[63] Benjamin Scellier. *A deep learning theory for neural networks grounded in physics*. PhD thesis, Université de Montréal, Quebec, Canada, 2020.

[64] Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pages 4644–4653. PMLR, 2018.

[65] P. Smolensky. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, page 194–281. MIT Press, Cambridge, MA, USA, 1986.

[66] Jiaming Song, Chenlin Meng, and Stefano Ermon. International conference on learning representations. 2021.

[67] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[68] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[69] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34: 1415–1428, 2021.

[70] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[71] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[72] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, Cambridge, UK, 2019.

[73] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[74] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[75] Binxu Wang and Carlos R Ponce. The geometry of deep generative image models and its applications. In *International Conference on Learning Representations*, 2021.

[76] Ying Nian Wu, Jianwen Xie, Yang Lu, and Song-Chun Zhu. Sparse and deep generalizations of the frame model. *Annals of Mathematical Sciences and Applications*, 3(1): 211–254, 2018.

[77] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *International Conference on Learning Representations*, 2022.

[78] Jianwen Xie, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Learning sparse FRAME models for natural image patterns. *International Journal of Computer Vision*, 114(2):91–112, 2015.

[79] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Inducing wavelets into random fields via generative boosting. *Applied and Computational Harmonic Analysis*, 41(1):4–25, 2016.

[80] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 7093–7101, 2017.

[81] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8629–8638, 2018.

[82] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via MCMC teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

[83] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):27–45, 2018b.

[84] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[85] Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, volume 2, 2021.

[86] Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint arXiv:1611.01799*, 2016.

[87] Song Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 847–854. IEEE, 1998.

[88] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

[89] Yaxuan Zhu, Jianwen Xie, Yingnian Wu, and Ruiqi Gao. Learning energy-based models by cooperative diffusion recovery likelihood. In *International Conference on Learning Representations*, 2024.